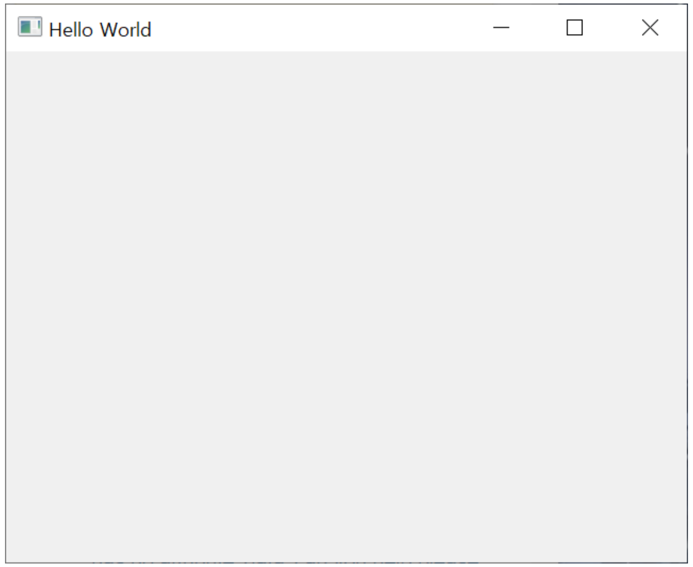
6. PyQt 프로그래밍

기본 위젯

```
import sys

from PyQt5.QtWidgets import QWidget
from PyQt5.QtWidgets import QApplication
from PyQt5.QtCore import Qt

class Form(QWidget):
    def __init__(self):
        QWidget.__init__(self, flags=Qt.Widget)
        self.init_widget()

    def init_widget(self):
        self.setWindowTitle("Hello World")

if __name__ == "__main__":
    app = QApplication(sys.argv)
    form = Form()
    form.show()
    exit(app.exec_())
```



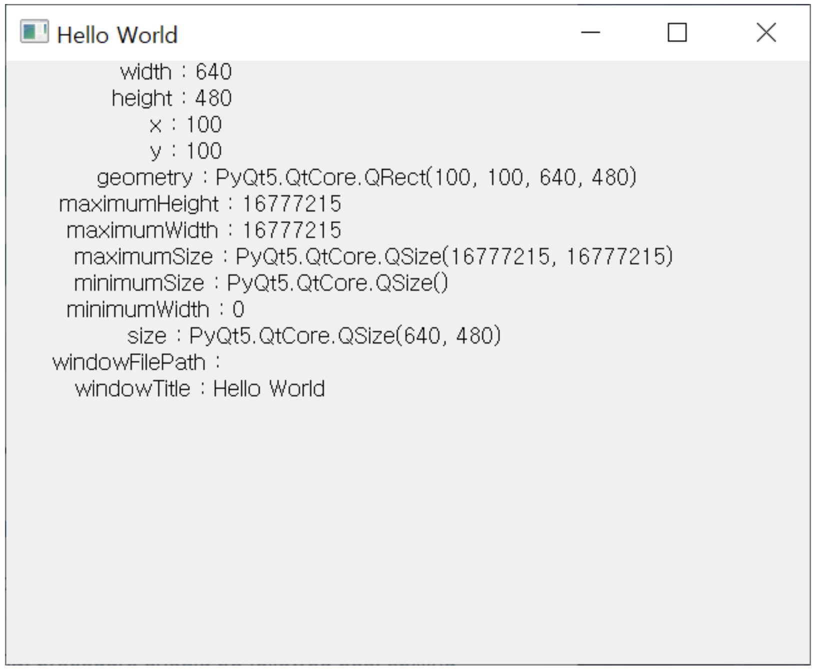
QWidget의 속성 값을 QLabel을 이용하여 출력하기

```python
import sys
from PyQt5.QtWidgets import QWidget
from PyQt5.QtWidgets import QLabel
from PyQt5.QtWidgets import QApplication
from PyQt5.QtCore import Qt

class Form(QWidget):
    def __init__(self):
        QWidget.__init__(self, flags=Qt.Widget)
        self.lb = None
        self.init_widget()

    def init_widget(self):
        self.setWindowTitle("Hello World")
        self.setGeometry(100, 100, 640, 480)
        self.lb = QLabel(self)
        properties_list = (
            "width", "height", "x", "y", "geometry",
            "maximumHeight", "maximumWidth", "maximumSize", "minimumSize", "minimumWidth",
            "size", "windowFilePath", "windowTitle"
        )
        msg = self.get_properties_value(properties_list)
        self.lb.setText(msg)
```

QWidget의 속성 값을 QLabel을 이용하여 출력하기

```
 def get_properties_value(self, properties):
        msg = []
        for p in properties:
            if not hasattr(self, p):
                continue
            value = getattr(self, p)()
            msg.append("{:>20s} : {:<30s}".format(p, str(value)))
        msg = "\n".join(msg)
        return msg


if __name__ == "__main__":
    app = QApplication(sys.argv)
    form = Form()
    form.show()
    exit(app.exec_())
```

QWidget의 속성 값을 QLabel을 이용하여 출력하기



```
           width : 640
          height : 480
               x : 100
               y : 100
        geometry : PyQt5.QtCore.QRect(100, 100, 640, 480)
   maximumHeight : 16777215
    maximumWidth : 16777215
     maximumSize : PyQt5.QtCore.QSize(16777215, 16777215)
     minimumSize : PyQt5.QtCore.QSize()
    minimumWidth : 0
            size : PyQt5.QtCore.QSize(640, 480)
  windowFilePath :
     windowTitle : Hello World
```

PaintEvent를 이용하여 실시간으로 값 변경하기

```
import sys

from PyQt5.QtWidgets import QWidget
from PyQt5.QtWidgets import QLabel
from PyQt5.QtWidgets import QApplication
from PyQt5.QtCore import Qt

class Form(QWidget):
    def __init__(self):
        QWidget.__init__(self, flags=Qt.Widget)
        self.mouse_x = 0
        self.mouse_y = 0

        self.lb = QLabel(self)

        self.properties_list = (
            "width", "height", "x", "y", "geometry",
            "maximumHeight", "maximumWidth", "maximumSize", "minimumSize", "minimumWidth",
            "size", "windowFilePath", "windowTitle",
            "underMouse"
        )

        self.init_widget()
```

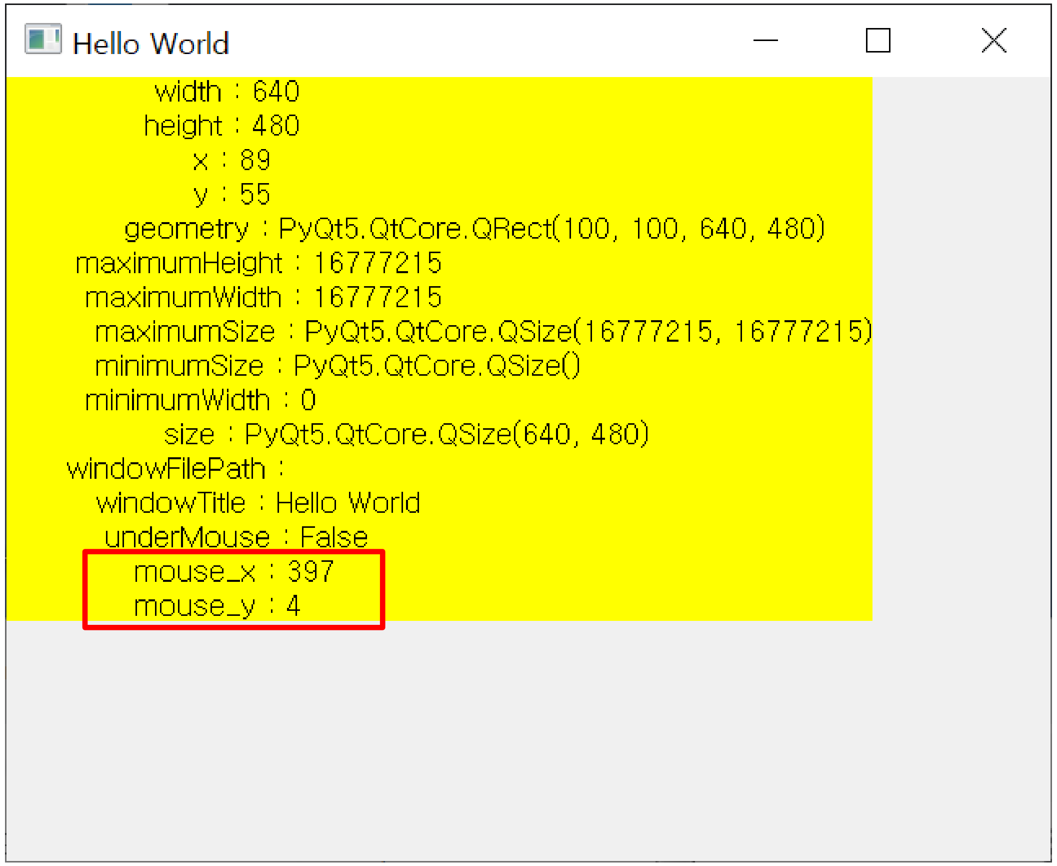
PaintEvent를 이용하여 실시간으로 값 변경하기

```
    def init_widget(self):
        self.setWindowTitle("Hello World")
        self.setGeometry(100, 100, 640, 480)
        self.setMouseTracking(True)

        self.lb.setStyleSheet("background-color: yellow")
        self.lb.setMouseTracking(True)
        msg = self.get_properties_value(self.properties_list)
        self.lb.setText(msg)

    def get_properties_value(self, properties):
        msg = []
        for p in properties:
            if not hasattr(self, p):
                continue
            value = getattr(self, p)()
            msg.append("{:>20s} : {:<30s}".format(p, str(value)))
        msg.append("{:>20s} : {:<30s}".format("mouse_x", str(self.mouse_x)))
        msg.append("{:>20s} : {:<30s}".format("mouse_y", str(self.mouse_y)))
        msg = "\n".join(msg)
        return msg
```

PaintEvent를 이용하여 실시간으로 값 변경하기

```
    def mouseMoveEvent(self, QMouseEvent):
        self.mouse_x = QMouseEvent.x()
        self.mouse_y = QMouseEvent.y()
        self.update()

    def moveEvent(self, QMoveEvent):
        self.update()

    def paintEvent(self, QPaintEvent):
        msg = self.get_properties_value(self.properties_list)
        self.lb.setText(msg)

if __name__ == "__main__":
    app = QApplication(sys.argv)
    form = Form()
    form.show()
    exit(app.exec_())
```

PaintEvent를 이용하여 실시간으로 값 변경하기



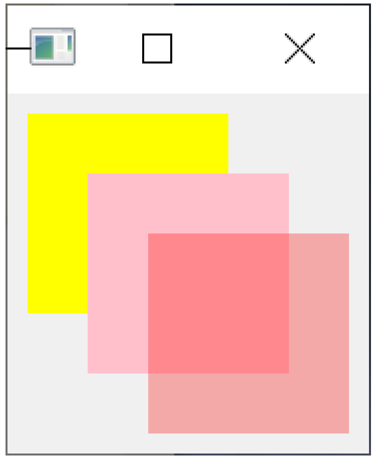
```
          width : 640
         height : 480
              x : 89
              y : 55
       geometry : PyQt5.QtCore.QRect(100, 100, 640, 480)
  maximumHeight : 16777215
   maximumWidth : 16777215
    maximumSize : PyQt5.QtCore.QSize(16777215, 16777215)
    minimumSize : PyQt5.QtCore.QSize()
   minimumWidth : 0
           size : PyQt5.QtCore.QSize(640, 480)
 windowFilePath :
    windowTitle : Hello World
     underMouse : False
        mouse_x : 397
        mouse_y : 4
```

위젯 투명도 설정

```
import sys

from PyQt5.QtWidgets import QWidget
from PyQt5.QtWidgets import QApplication
from PyQt5.QtCore import Qt
from PyQt5.QtWidgets import QGraphicsOpacityEffect

class Form(QWidget):
    def __init__(self):
        QWidget.__init__(self, flags=Qt.Widget)

        self.w1 = QWidget(parent=self, flags=Qt.Widget)
        self.w3 = QWidget(parent=self, flags=Qt.Widget)
        self.w2 = QWidget(parent=self, flags=Qt.Widget)

        self.init_widget()
```

위젯 투명도 설정

```
 def init_widget(self):
        self.setWindowTitle("Hello World")

        self.w1.setGeometry(10, 10, 100, 100)
        self.w1.setStyleSheet("background-color: yellow")

        self.w3.setGeometry(40, 40, 100, 100)
        self.w3.setStyleSheet("background-color: pink")

        self.w2.setGeometry(70, 70, 100, 100)
        opacity_effect = QGraphicsOpacityEffect(self.w2)
        opacity_effect.setOpacity(0.3)
        self.w2.setGraphicsEffect(opacity_effect)
        self.w2.setStyleSheet("background-color: red")

if __name__ == "__main__":
    app = QApplication(sys.argv)
    form = Form()
    form.show()
    exit(app.exec_())
```



마우스의 움직임에 따라 위젯의 배경색을 바꾼다

```
import sys

from PyQt5.QtWidgets import QWidget
from PyQt5.QtWidgets import QLabel
from PyQt5.QtWidgets import QApplication
from PyQt5.QtCore import Qt
from PyQt5.QtGui import QColor
from PyQt5.QtGui import QPalette

from math import sqrt

class Label(QLabel):
    def __init__(self, parent=None):
        super(Label, self).__init__(parent)
        self.setMouseTracking(True)
        self.setAutoFillBackground(True)
        self.rgb = str()
```

마우스의 움직임에 따라 위젯의 배경색을 바꾼다

```
class Form(QWidget):
    def __init__(self):
        QWidget.__init__(self, flags=Qt.Widget)

        self.mouse_x = 0
        self.mouse_y = 0

        self.lb = Label(self)
        self.properties_list = (
            "width", "height", "x", "y", "geometry",
            "maximumHeight", "maximumWidth", "maximumSize", "minimumSize", "minimumWidth",
            "size", "windowFilePath", "windowTitle",
            "underMouse"
        )
        self.rgb_value = ''

        self.init_widget()
        self.show()
```

마우스의 움직임에 따라 위젯의 배경색을 바꾼다

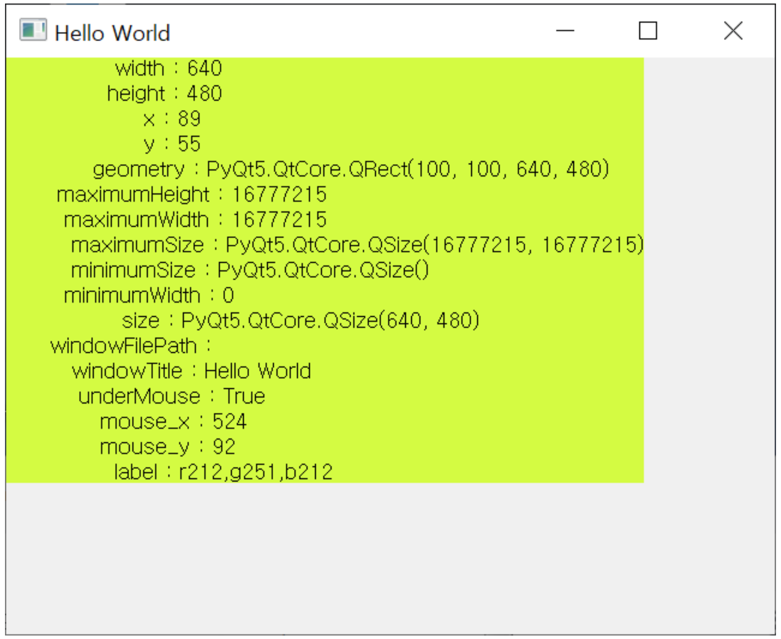
```
def init_widget(self):
        self.setWindowTitle("Hello World")
        self.setGeometry(100, 100, 640, 480)
        self.setMouseTracking(True)

        msg = self.get_properties_value(self.properties_list)
        self.lb.setText(msg)

    def get_properties_value(self, properties):
        msg = []
        for p in properties:
            if not hasattr(self, p):
                continue
            value = getattr(self, p)()
            msg.append("{:>20s} : {:<30s}".format(p, str(value)))
        msg.append("{:>20s} : {:<30s}".format("mouse_x", str(self.mouse_x)))
        msg.append("{:>20s} : {:<30s}".format("mouse_y", str(self.mouse_y)))
        msg.append("{:>20s} : {:<30s}".format("label", self.rgb_value))
        msg = "\n".join(msg)
        return msg
```

마우스의 움직임에 따라 위젯의 배경색을 바꾼다

```
    def mouseMoveEvent(self, QMouseEvent):
        self.mouse_x = QMouseEvent.x()
        self.mouse_y = QMouseEvent.y()
        self.set_widget_bgcolor(self.lb, self.mouse_x, self.mouse_y)
        self.update()

    def set_widget_bgcolor(self, widget, mx, my):
        if not isinstance(widget, QWidget):
            return False
        lw = widget.width()
        lh = widget.height()
        if (lw < mx) or (lh <my):
            return False

        v = int(sqrt(mx ** 2 + my ** 2))
        r = int(sqrt(lw ** 2 + lh ** 2))
        r = int(v/r * 255)
        g = int(mx / lw * 255)
        b = int(my / lh * 255)
        pal = QPalette()
        pal.setColor(QPalette.Background, QColor(r, g, b))
        widget.setPalette(pal)
        self.rgb_value = "r{0},g{1},b{0}".format(r, g, b)
```

마우스의 움직임에 따라 위젯의 배경색을 바꾼다

```
 def moveEvent(self, QMoveEvent):
        self.update()

    def paintEvent(self, QPaintEvent):
        msg = self.get_properties_value(self.properties_list)
        self.lb.setText(msg)

if __name__ == "__main__":
    app = QApplication(sys.argv)
    form = Form()
    exit(app.exec_())
```



Hello World

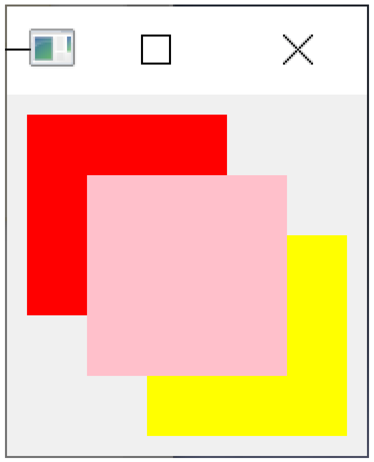
```
        width : 640
       height : 480
            x : 89
            y : 55
     geometry : PyQt5.QtCore.QRect(100, 100, 640, 480)
maximumHeight : 16777215
 maximumWidth : 16777215
  maximumSize : PyQt5.QtCore.QSize(16777215, 16777215)
  minimumSize : PyQt5.QtCore.QSize()
 minimumWidth : 0
         size : PyQt5.QtCore.QSize(640, 480)
windowFilePath :
  windowTitle : Hello World
   underMouse : True
      mouse_x : 524
      mouse_y : 92
        label : r212,g251,b212
```

겹쳐진 위젯의 순서 변경

```
import sys

from PyQt5.QtWidgets import QWidget
from PyQt5.QtWidgets import QLabel
from PyQt5.QtWidgets import QApplication
from PyQt5.QtCore import Qt

class StackWidget(QLabel):
    def __init__(self, color, parent=None):
        super(StackWidget, self).__init__(parent)
        self.setStyleSheet("background-color: %s" % color)

    def mousePressEvent(self, event):
        if not self.underMouse():
            return
        self.raise_()
```

겹쳐진 위젯의 순서 변경

```
class Form(QWidget):
    def __init__(self):
        QWidget.__init__(self, flags=Qt.Widget)
        self.setWindowTitle("Widget raise and lower")

        self.lb1 = StackWidget("red", parent=self)
        self.lb1.setGeometry(10, 10, 100, 100)

        self.lb2 = StackWidget("pink", parent=self)
        self.lb2.setGeometry(40, 40, 100, 100)

        self.lb3 = StackWidget("yellow", parent=self)
        self.lb3.setGeometry(70, 70, 100, 100)

if __name__ == "__main__":
    app = QApplication(sys.argv)
    form = Form()
    form.show()
    exit(app.exec_())
```



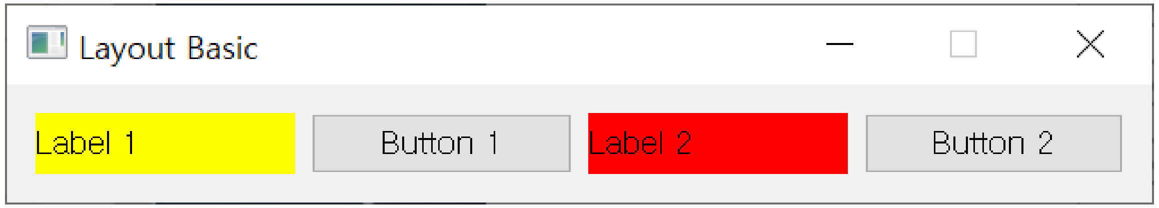
6. PyQt 프로그래밍

레이아웃 사용 기본

```
import sys

from PyQt5.QtWidgets import QWidget
from PyQt5.QtWidgets import QLabel
from PyQt5.QtWidgets import QPushButton
from PyQt5.QtWidgets import QBoxLayout

from PyQt5.QtWidgets import QApplication
from PyQt5.QtCore import Qt

class Form(QWidget):
    def __init__(self):
        QWidget.__init__(self, flags=Qt.Widget)
        self.lb_1 = QLabel()
        self.lb_2 = QLabel()
        self.pb_1 = QPushButton()
        self.pb_2 = QPushButton()
        self.layout_1 = QBoxLayout(QBoxLayout.LeftToRight, self)
        self.setLayout(self.layout_1)
        self.init_widget()
```

레이아웃 사용 기본

```
 def init_widget(self):
        self.setWindowTitle("Layout Basic")
        self.setFixedWidth(640)

        self.lb_1.setText("Label 1")
        self.lb_1.setStyleSheet("background-color: yellow")
        self.pb_1.setText("Button 1")
        self.layout_1.addWidget(self.lb_1)
        self.layout_1.addWidget(self.pb_1)

        self.lb_2.setText("Label 2")
        self.lb_2.setStyleSheet("background-color: red")
        self.pb_2.setText("Button 2")
        self.layout_1.addWidget(self.lb_2)
        self.layout_1.addWidget(self.pb_2)

if __name__ == "__main__":
    app = QApplication(sys.argv)
    form = Form()
    form.show()
    exit(app.exec_())
```



레이아웃에 배치된 위젯 정렬 1

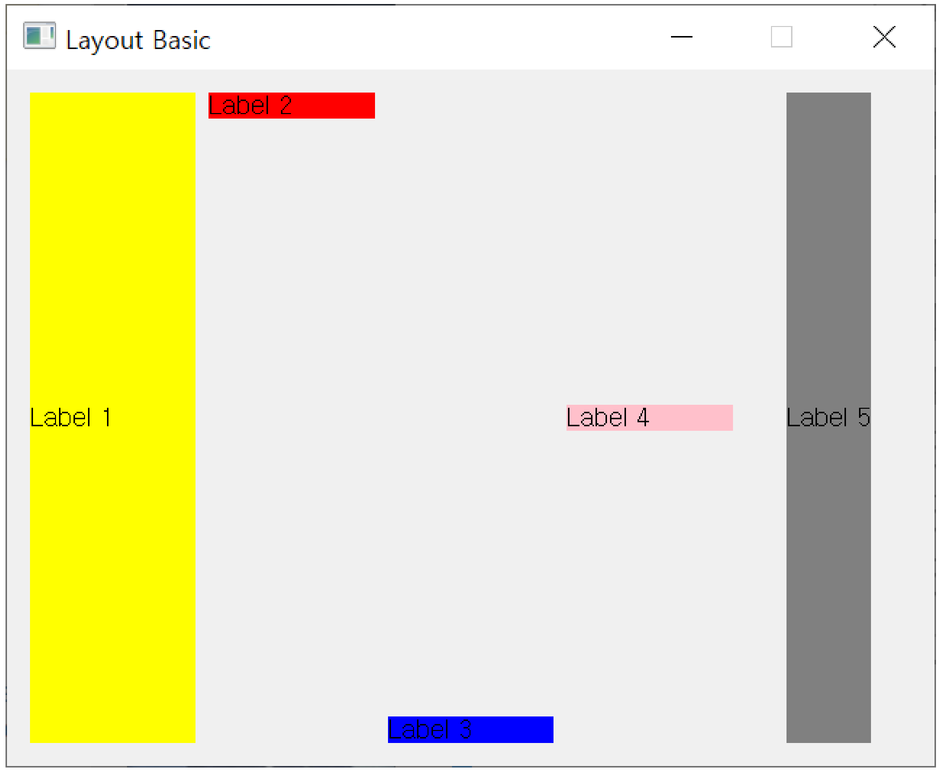
```
import sys

from PyQt5.QtWidgets import QWidget
from PyQt5.QtWidgets import QLabel
from PyQt5.QtWidgets import QBoxLayout

from PyQt5.QtWidgets import QApplication
from PyQt5.QtCore import Qt

class Form(QWidget):
    def __init__(self):
        QWidget.__init__(self, flags=Qt.Widget)

        self.lb_1 = QLabel()
        self.lb_2 = QLabel()
        self.lb_3 = QLabel()
        self.lb_4 = QLabel()
        self.lb_5 = QLabel()

        self.layout_1 = QBoxLayout(QBoxLayout.LeftToRight, self)
        self.setLayout(self.layout_1)
        self.init_widget()
```

레이아웃에 배치된 위젯 정렬 1

```
    def init_widget(self):
        self.setWindowTitle("Layout Basic")
        self.setFixedWidth(640)
        self.setFixedHeight(480)

        self.lb_1.setText("Label 1")
        self.lb_2.setText("Label 2")
        self.lb_3.setText("Label 3")
        self.lb_4.setText("Label 4")
        self.lb_5.setText("Label 5")

        self.lb_1.setStyleSheet("background-color: yellow")
        self.lb_2.setStyleSheet("background-color: red")
        self.lb_3.setStyleSheet("background-color: blue")
        self.lb_4.setStyleSheet("background-color: pink")
        self.lb_5.setStyleSheet("background-color: grey")

        self.layout_1.addWidget(self.lb_1)
        self.layout_1.addWidget(self.lb_2, alignment=Qt.AlignTop)
        self.layout_1.addWidget(self.lb_3, alignment=Qt.AlignBottom)
        self.layout_1.addWidget(self.lb_4, alignment=Qt.AlignVCenter)
        self.layout_1.addWidget(self.lb_5, alignment=Qt.AlignHCenter)
```

레이아웃에 배치된 위젯 정렬 1

```
if __name__ == "__main__":
    app = QApplication(sys.argv)
    form = Form()
    form.show()
    exit(app.exec_())
```



레이아웃에 배치된 위젯 정렬 2

```
import sys

from PyQt5.QtWidgets import QWidget
from PyQt5.QtWidgets import QLabel
from PyQt5.QtWidgets import QBoxLayout

from PyQt5.QtWidgets import QApplication
from PyQt5.QtCore import Qt
```

레이아웃에 배치된 위젯 정렬 2

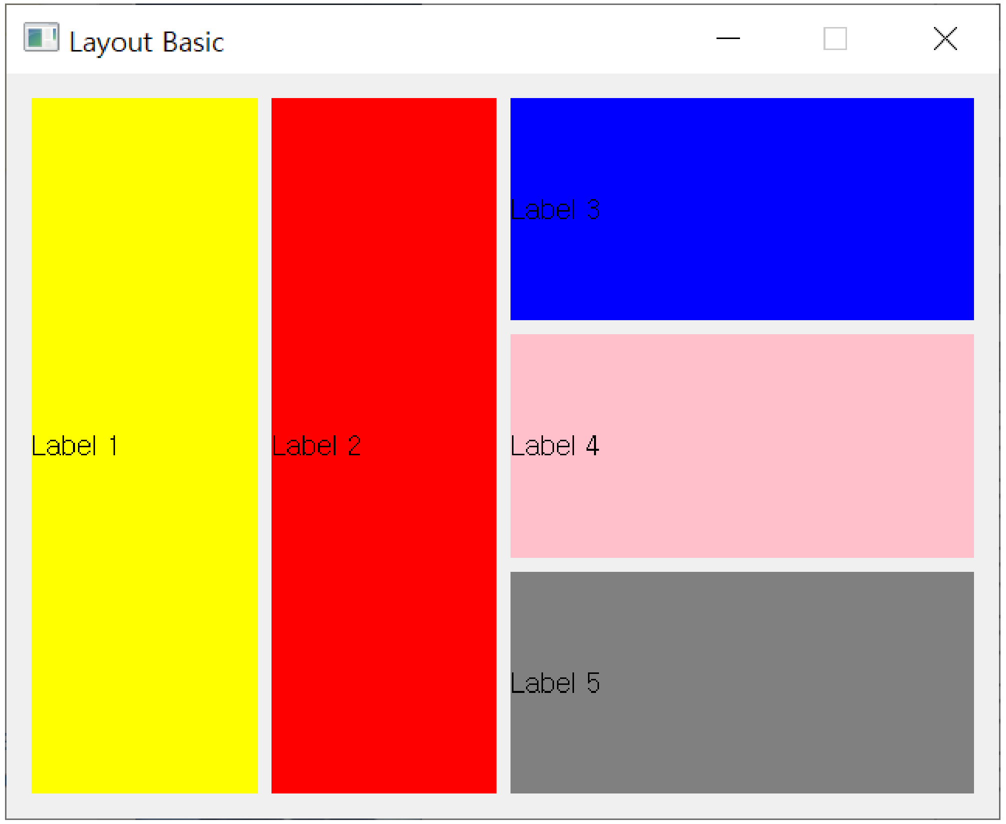
```
class Form(QWidget):
    def __init__(self):
        QWidget.__init__(self, flags=Qt.Widget)

        self.lb_1 = QLabel()
        self.lb_2 = QLabel()
        self.lb_3 = QLabel()
        self.lb_4 = QLabel()
        self.lb_5 = QLabel()

        self.layout_1 = QBoxLayout(QBoxLayout.LeftToRight, self)
        self.layout_2 = QBoxLayout(QBoxLayout.LeftToRight)
        self.layout_3 = QBoxLayout(QBoxLayout.TopToBottom)

        self.layout_1.addLayout(self.layout_2)
        self.layout_1.addLayout(self.layout_3)

        self.setLayout(self.layout_1)
        self.init_widget()
```

레이아웃에 배치된 위젯 정렬 2

```
    def init_widget(self):
        self.setWindowTitle("Layout Basic")
        self.setFixedWidth(640)
        self.setFixedHeight(480)

        self.lb_1.setText("Label 1")
        self.lb_2.setText("Label 2")
        self.lb_3.setText("Label 3")
        self.lb_4.setText("Label 4")
        self.lb_5.setText("Label 5")

        self.lb_1.setStyleSheet("background-color: yellow")
        self.lb_2.setStyleSheet("background-color: red")
        self.lb_3.setStyleSheet("background-color: blue")
        self.lb_4.setStyleSheet("background-color: pink")
        self.lb_5.setStyleSheet("background-color: grey")

        self.layout_2.addWidget(self.lb_1)
        self.layout_2.addWidget(self.lb_2)
        self.layout_3.addWidget(self.lb_3)
        self.layout_3.addWidget(self.lb_4)
        self.layout_3.addWidget(self.lb_5)
```

레이아웃에 배치된 위젯 정렬 2

```
if __name__ == "__main__":
    app = QApplication(sys.argv)
    form = Form()
    form.show()
    exit(app.exec_())
```



다양한 레이아웃 위젯 사용

```
import sys

from PyQt5.QtWidgets import QWidget
from PyQt5.QtWidgets import QLabel
from PyQt5.QtWidgets import QSpacerItem
from PyQt5.QtWidgets import QLineEdit
from PyQt5.QtWidgets import QTextEdit
from PyQt5.QtWidgets import QPushButton
from PyQt5.QtWidgets import QGroupBox

from PyQt5.QtWidgets import QBoxLayout
from PyQt5.QtWidgets import QHBoxLayout
from PyQt5.QtWidgets import QGridLayout
from PyQt5.QtWidgets import QFormLayout

from PyQt5.QtWidgets import QApplication
from PyQt5.QtCore import Qt
```

다양한 레이아웃 위젯 사용

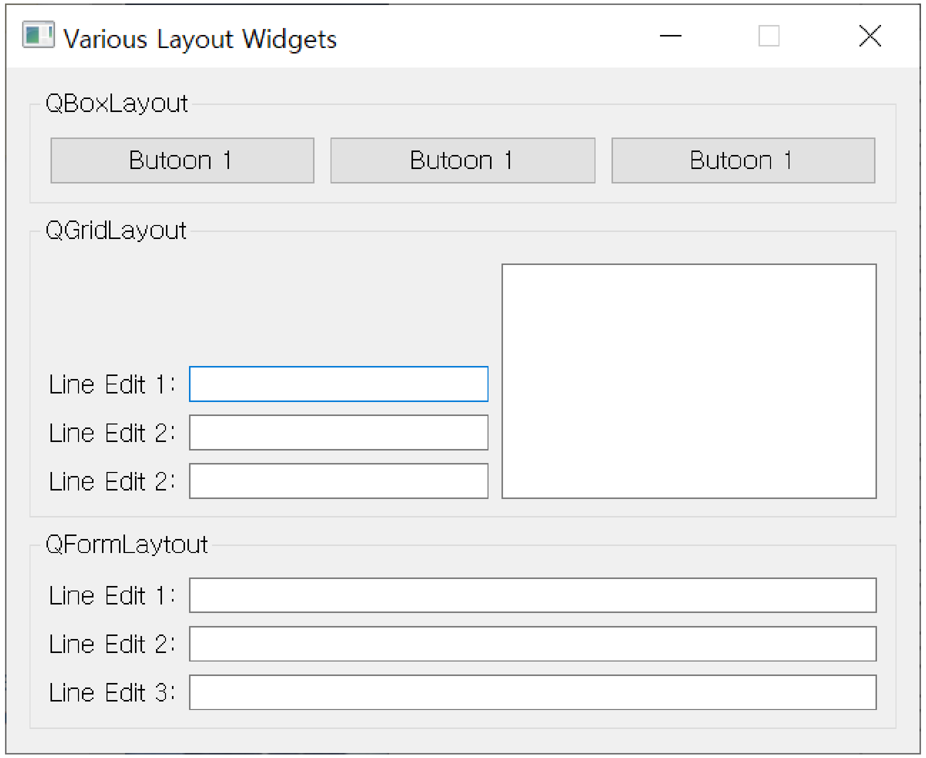
```
class Form(QWidget):
    def __init__(self):
        QWidget.__init__(self, flags=Qt.Widget)

        self.setWindowTitle("Various Layout Widgets")
        self.setFixedWidth(640)
        self.setFixedHeight(480)

        layout_base = QBoxLayout(QBoxLayout.TopToBottom, self)
        self.setLayout(layout_base)

        grp_1 = QGroupBox("QBoxLayout")
        layout_base.addWidget(grp_1)
        layout = QHBoxLayout()
        layout.addWidget(QPushButton("Butoon 1"))
        layout.addWidget(QPushButton("Butoon 1"))
        layout.addWidget(QPushButton("Butoon 1"))
        grp_1.setLayout(layout)
```

다양한 레이아웃 위젯 사용

```
grp_2 = QGroupBox("QGridLayout")
    layout_base.addWidget(grp_2)
    grp_2_layout = QBoxLayout(QBoxLayout.LeftToRight)
    grp_2.setLayout(grp_2_layout)
    layout = QGridLayout()
    layout.addItem(QSpacerItem(10, 100))
    layout.addWidget(QLabel("Line Edit 1:"), 1, 0)
    layout.addWidget(QLabel("Line Edit 2:"), 2, 0)
    layout.addWidget(QLabel("Line Edit 2:"), 3, 0)
    layout.addWidget(QLineEdit(), 1, 1)
    layout.addWidget(QLineEdit(), 2, 1)
    layout.addWidget(QLineEdit(), 3, 1)
    grp_2_layout.addLayout(layout)
    grp_2_layout.addWidget(QTextEdit())

    grp_3 = QGroupBox("QFormLaytout")
    layout_base.addWidget(grp_3)
    layout = QFormLayout()
    grp_3.setLayout(layout)
    layout.addRow(QLabel("Line Edit 1:"), QLineEdit())
    layout.addRow(QLabel("Line Edit 2:"), QLineEdit())
    layout.addRow(QLabel("Line Edit 3:"), QLineEdit())
```

다양한 레이아웃 위젯 사용

```
if __name__ == "__main__":
    app = QApplication(sys.argv)
    form = Form()
    form.show()
    exit(app.exec_())
```



6. PyQt 프로그래밍

Signal과 Slot 연결 기본

QDial 위젯은 몇 가지 시그널을 갖고 있다.

여기서는 valueChanged 시그널을 QSlider 슬롯에 연결한다. 슬롯은 숫자를 받아서 QSlider 위젯에 표시하는 역할을 한다.

여기서 시그널을 보내는 객체인 송신자 (sender)는 dial, 시그널을 받는 객체인 수신자 (receiver)는 QSlider이다.

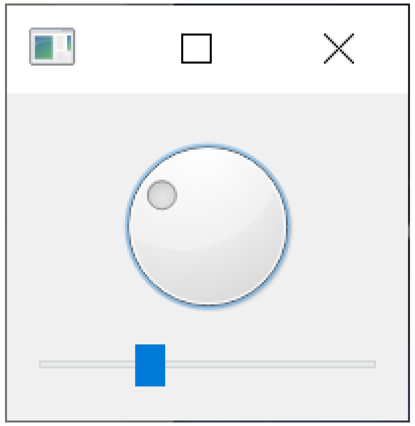
슬롯 (slot)은 시그널에 어떻게 반응할지를 구현한 메서드이다.

```
import sys

from PyQt5.QtWidgets import QWidget
from PyQt5.QtWidgets import QDial
from PyQt5.QtWidgets import QSlider
from PyQt5.QtWidgets import QApplication
from PyQt5.QtWidgets import QBoxLayout
from PyQt5.QtCore import Qt
```

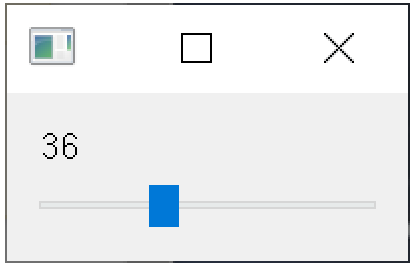
Signal과 Slot 연결 기본

```
class Form(QWidget):
    def __init__(self):
        QWidget.__init__(self, flags=Qt.Widget)
        self.dl = QDial()
        self.sd = QSlider(Qt.Horizontal)
        self.init_widget()

    def init_widget(self):
        self.setWindowTitle("Signal Slot")
        form_lbx = QBoxLayout(QBoxLayout.TopToBottom, parent=self)
        self.setLayout(form_lbx)

        self.dl.valueChanged.connect(self.sd.setValue)
        self.sd.valueChanged.connect(self.dl.setValue)

        form_lbx.addWidget(self.dl)
        form_lbx.addWidget(self.sd)

if __name__ == "__main__":
    app = QApplication(sys.argv)
    form = Form()
    form.show()
    exit(app.exec_())
```

Signal과 Slot 연결 기본



Signal 과 Slot 람다 함수를 이용하여 값 처리 및 전달



QSlider의 값을 QLabel에 표시하고 싶다.

문제 :
QLabel.setText는 문자열만 받으므로 정수형을 주는
QSlider.valueChange를 바로 사용할 수 없다.

해결:
이를 해결하기 위해선 따로 처리해주는 함수를 만들어 줘야하는데
lambda를 이용하여 값을 넘기는 방법을 사용하면 간편하게 해결할 수
있다.

```
import sys

from PyQt5.QtWidgets import QWidget
from PyQt5.QtWidgets import QLabel
from PyQt5.QtWidgets import QSlider
from PyQt5.QtWidgets import QApplication
from PyQt5.QtWidgets import QBoxLayout
from PyQt5.QtCore import Qt
```

Signal 과 Slot 람다 함수를 이용하여 값 처리 및 전달

```
class Form(QWidget):
    def __init__(self):
        QWidget.__init__(self, flags=Qt.Widget)
        self.lb = QLabel()
        self.sd = QSlider(Qt.Horizontal)
        self.init_widget()

    def init_widget(self):
        self.setWindowTitle("Signal Slot")
        form_lbx = QBoxLayout(QBoxLayout.TopToBottom, parent=self)
        self.setLayout(form_lbx)

        self.sd.valueChanged.connect(
            lambda v: self.lb.setText(str(v))
        )

        form_lbx.addWidget(self.lb)
        form_lbx.addWidget(self.sd)

if __name__ == "__main__":
    app = QApplication(sys.argv)
    form = Form()
    form.show()
    exit(app.exec_())
```

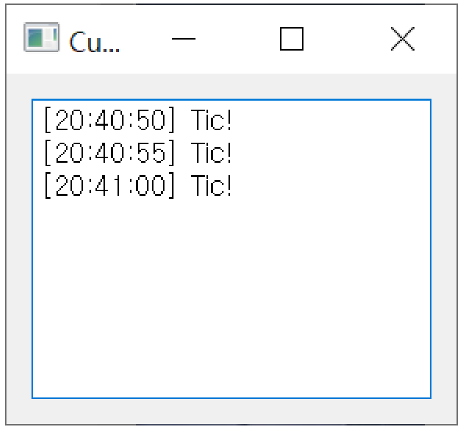
Signal 과 Slot 사용자정의 시그널 만들기

아래 모듈을 임포트 한다.
from PyQt5.QtCore import pyqtSignal

Tic이란 이름의 시그널을 생성한다.
tic = pyqtSignal(name="Tic")

시그널을 슬롯에게 람다를 통해 전달 한다.

```
self.tic_gen.Tic.connect(
        lambda: self.te.insertPlainText(time.strftime("[%H:%M:%S] Tic!₩n"))
    )
```



```
[20:40:50] Tic!
[20:40:55] Tic!
[20:41:00] Tic!
```

```
import sys
import time

from PyQt5.QtWidgets import QWidget
from PyQt5.QtWidgets import QTextEdit
from PyQt5.QtWidgets import QApplication
from PyQt5.QtWidgets import QBoxLayout
from PyQt5.QtCore import Qt
from PyQt5.QtCore import QThread
from PyQt5.QtCore import pyqtSignal
```

Signal 과 Slot 사용자정의 시그널 만들기

```
class TicGenerator(QThread):
    tic = pyqtSignal(name="Tic")

    def __init__(self):
        QThread.__init__(self)

    def __del__(self):
        self.wait()

    def run(self):
        while True:
            t = int(time.time())
            if not t % 5 == 0:
                self.usleep(1)
                continue
            self.Tic.emit()
            self.msleep(1000)
```

Signal 과 Slot 사용자정의 시그널 만들기

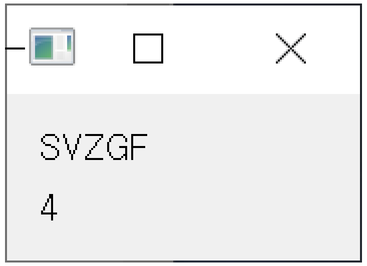
```
class Form(QWidget):
    def __init__(self):
        QWidget.__init__(self, flags=Qt.Widget)
        self.te = QTextEdit()
        self.tic_gen = TicGenerator()
        self.init_widget()
        self.tic_gen.start()

    def init_widget(self):
        self.setWindowTitle("Custom Signal")
        form_lbx = QBoxLayout(QBoxLayout.TopToBottom, parent=self)
        self.setLayout(form_lbx)

        self.tic_gen.Tic.connect(
            lambda: self.te.insertPlainText(time.strftime("[%H:%M:%S] Tic!\n"))
        )

        form_lbx.addWidget(self.te)

if __name__ == "__main__":
    app = QApplication(sys.argv)
    form = Form()
    form.show()
    exit(app.exec_())
```

Signal 과 Slot 사용자정의 시그널 반환 값 타입

시그널이 두개 이상 서로 다른 타입으로 전송하고 싶다.

OTP에 해당 하는 시그널은 문자열로 전달한다.
value_changed = pyqtSignal(str, name="ValueChanged")

1초마다 남은 시간은 int로 전달한다.
expires_in = pyqtSignal(int, name="ExpiresIn")


```
SVZGF
4
```

```
import sys

from PyQt5.QtWidgets import QWidget
from PyQt5.QtWidgets import QLabel
from PyQt5.QtWidgets import QApplication
from PyQt5.QtWidgets import QBoxLayout
from PyQt5.QtCore import Qt
from PyQt5.QtCore import QThread
from PyQt5.QtCore import pyqtSignal
import string
import time
import random
```

Signal 과 Slot 사용자정의 시그널 반환 값 타입

```
class OtpTokenGenerator(QThread):
    value_changed = pyqtSignal(str, name="ValueChanged")
    expires_in = pyqtSignal(int, name="ExpiresIn")

    EXPIRE_TIME = 5

    def __init__(self):
        QThread.__init__(self)
        self.characters = list(string.ascii_uppercase)
        self.token = self.generate()

    def __del__(self):
        self.wait()

    def generate(self):
        random.shuffle(self.characters)
        return ''.join(self.characters[0:5])
```

Signal 과 Slot 사용자정의 시그널 반환 값 타입

```
    def run(self):
        self.value_changed.emit(self.token)
        while True:
            t = int(time.time()) % self.EXPIRE_TIME
            self.expires_in.emit(self.EXPIRE_TIME - t)
            if t != 0:
                self.usleep(1)
                continue

            self.token = self.generate()
            self.value_changed.emit(self.token)
            self.msleep(1000)
```

Signal 과 Slot 사용자정의 시그널 반환 값 타입

```
class Form(QWidget):
    def __init__(self):
        QWidget.__init__(self, flags=Qt.Widget)
        self.lb_token = QLabel()
        self.lb_expire_time = QLabel()
        self.otp_gen = OtpTokenGenerator()
        self.init_widget()
        self.otp_gen.start()

    def init_widget(self):
        self.setWindowTitle("Custom Signal")
        form_lbx = QBoxLayout(QBoxLayout.TopToBottom, parent=self)
        self.setLayout(form_lbx)

        self.otp_gen.ValueChanged.connect(self.lb_token.setText)
        self.otp_gen.ExpiresIn.connect(lambda v: self.lb_expire_time.setText(str(v)))

        form_lbx.addWidget(self.lb_token)
        form_lbx.addWidget(self.lb_expire_time)
```

Signal 과 Slot 사용자정의 시그널 반환 값 타입

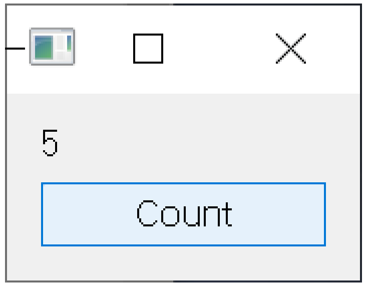
```
if __name__ == "__main__":
    app = QApplication(sys.argv)
    form = Form()
    form.show()
    exit(app.exec_())
```

Signal 과 Slot 사용자정의 슬롯 만들기

값을 유지하는 사용자 정의 슬롯을 만들고 싶다.

아래와 같이 함수를 만들고

```
@pyqtSlot()
    def count(self):
        self.cnt += 1
        self.lb.setText(str(self.cnt))
```



아래와 같이 호출 하면 된다.

```
self.pb.clicked.connect(self.count)
```

```
import sys

from PyQt5.QtWidgets import QWidget
from PyQt5.QtWidgets import QLabel
from PyQt5.QtWidgets import QPushButton
from PyQt5.QtWidgets import QApplication
from PyQt5.QtWidgets import QBoxLayout
from PyQt5.QtCore import Qt
from PyQt5.QtCore import pyqtSlot
```

Signal 과 Slot 사용자정의 슬롯 만들기

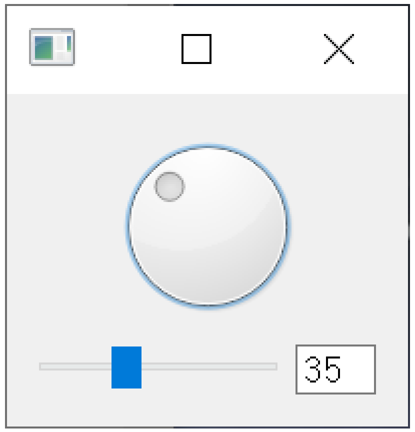
```
class Form(QWidget):
    def __init__(self):
        QWidget.__init__(self, flags=Qt.Widget)
        self.cnt = 0
        self.lb = QLabel(str(self.cnt))
        self.pb = QPushButton("Count")
        self.init_widget()

    def init_widget(self):
        self.setWindowTitle("Custom Signal")
        form_lbx = QBoxLayout(QBoxLayout.TopToBottom, parent=self)
        self.setLayout(form_lbx)
        self.pb.clicked.connect(self.count)

        form_lbx.addWidget(self.lb)
        form_lbx.addWidget(self.pb)

    @pyqtSlot()
    def count(self):
        self.cnt += 1
        self.lb.setText(str(self.cnt))
```

Signal 과 Slot 사용자정의 슬롯 만들기

```
if __name__ == "__main__":
    app = QApplication(sys.argv)
    form = Form()
    form.show()
    exit(app.exec_())
```

여러 타입의 값을 받을 수 있게 만들기

```
import sys

from PyQt5.QtWidgets import QWidget
from PyQt5.QtWidgets import QDial
from PyQt5.QtWidgets import QSlider
from PyQt5.QtWidgets import QLineEdit
from PyQt5.QtWidgets import QApplication
from PyQt5.QtWidgets import QBoxLayout
from PyQt5.QtCore import Qt
from PyQt5.QtCore import pyqtSlot

class CustomSlider(QSlider):
    def __init__(self, *args):
        QSlider.__init__(self, *args)

    @pyqtSlot(int)
    @pyqtSlot(str)
    def setValue(self, value):
        value = int(value)
        QSlider.setValue(self, value)
```



여러 타입의 값을 받을 수 있게 만들기

```python
class Form(QWidget):
    def __init__(self):
        QWidget.__init__(self, flags=Qt.Widget)
        self.cnt = 0
        self.le = QLineEdit()
        self.dial = QDial()
        self.sld = CustomSlider(Qt.Horizontal)
        self.init_widget()

    def init_widget(self):
        self.setWindowTitle("Custom Slot")
        form_lbx = QBoxLayout(QBoxLayout.TopToBottom, parent=self)
        control_lbx = QBoxLayout(QBoxLayout.LeftToRight, parent=self)
        self.setLayout(form_lbx)
        self.le.setMaximumWidth(40)
        self.sld.valueChanged.connect(self.valueChanged)
        self.le.textChanged.connect(self.sld.setValue)
        self.dial.valueChanged.connect(self.sld.setValue)

        form_lbx.addWidget(self.dial)
        form_lbx.addLayout(control_lbx)
        control_lbx.addWidget(self.sld)
        control_lbx.addWidget(self.le)
```

여러 타입의 값을 받을 수 있게 만들기

```
    @pyqtSlot(int, name="valueChanged")
    def value_changed(self, value):
        self.le.setText(str(value))
        self.dial.setValue(value)

if __name__ == "__main__":
    app = QApplication(sys.argv)
    form = Form()
    form.show()
    exit(app.exec_())
```

6. PyQt 프로그래밍

MVC (Model-View-Controller) 디자인 패턴을 사용하여 계산기를 구현한다. 이 패턴에는 각각 다른 역할을 가진 세 개의 코드 계층이 있다.

1. 이 모델 은 앱의 비즈니스 로직을 관리 한다.
   핵심 기능과 데이터를 포함한다.
   계산기의 경우 모델이 계산을 처리한다.

2. 뷰 는 앱의 GUI를 구현한다.
   최종 사용자가 응용 프로그램과 상호 작용하는 데 필요한 모든 위젯을 호스팅한다.
   이 뷰는 또한 사용자 작업 및 이벤트를 받는다.
   계산기의 경우 보기는 화면에 표시되는 창이다.

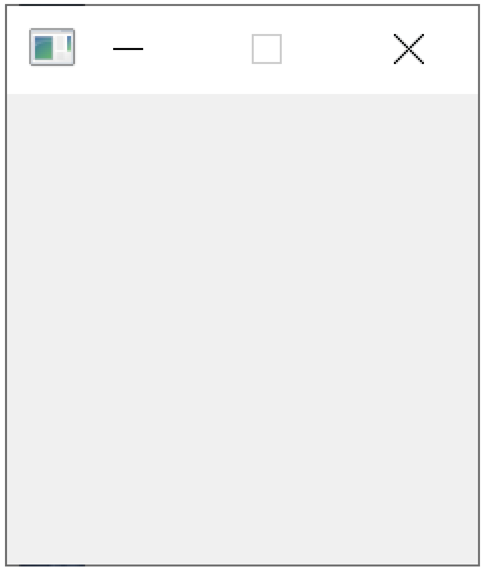
3. 컨트롤러 는 모델과 뷰를 연결하여 응용 프로그램을 작동시킨다.
   사용자 이벤트 (또는 요청)가 컨트롤러로 전송되어 모델이 작동한다.
   모델이 요청 된 결과 (또는 데이터)를 올바른 형식으로 전달하면
   컨트롤러가 결과를 보기로 전달한다.
   계산기의 경우 컨트롤러는 GUI에서 사용자 이벤트를 수신하고
   모델에게 계산을 수행하도록 요청하고 결과로 GUI를 업데이트 한다.

GUI 데스크탑 응용 프로그램을위한 단계별 MVC 패턴

1. 사용자는보기 (GUI)에서 조치 또는 요청 (이벤트)을 수행한다.
2. 보기는 사용자의 조치에 대해 컨트롤러에 알린다.
3. 컨트롤러는 사용자의 요청을 받고 응답을 위해 모델을 쿼리한다.
4. 모델은 컨트롤러 쿼리를 처리하고 필요한 작업을 수행하며 응답 또는 결과를 반환한다.
5. 컨트롤러는 모델의 답변을 받고 그에 따라 보기를 업데이트 한다.
6. 사용자는 최종적으로 보기에서 요청 된 결과를 본다.

스켈레톤 만들기

```
import sys

from PyQt5.QtWidgets import QApplication
from PyQt5.QtWidgets import QMainWindow
from PyQt5.QtWidgets import QWidget

class PyCalcUi(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle('PyCalc')
        self.setFixedSize(235, 235)
        self._centralWidget = QWidget(self)
        self.setCentralWidget(self._centralWidget)

def main():
    pycalc = QApplication(sys.argv)
    view = PyCalcUi()
    view.show()
    sys.exit(pycalc.exec_())

if __name__ == '__main__':
    main()
```



뷰 완성

```
import sys
from PyQt5.QtCore import Qt
from PyQt5.QtWidgets import QGridLayout
from PyQt5.QtWidgets import QLineEdit
from PyQt5.QtWidgets import QPushButton
from PyQt5.QtWidgets import QVBoxLayout
from PyQt5.QtWidgets import QApplication
from PyQt5.QtWidgets import QMainWindow
from PyQt5.QtWidgets import QWidget
from functools import partial
```

뷰 완성

```
class PyCalcUi(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle('PyCalc')
        self.setFixedSize(235, 235)
        self.generalLayout = QVBoxLayout()
        self._centralWidget = QWidget(self)
        self.setCentralWidget(self._centralWidget)
        self._centralWidget.setLayout(self.generalLayout)
        self._createDisplay()
        self._createButtons()
```

뷰 완성

```
def _createDisplay(self):
       self.display = QLineEdit()
       self.display.setFixedHeight(35)
       self.display.setAlignment(Qt.AlignRight)
       self.display.setReadOnly(True)
       self.generalLayout.addWidget(self.display)
```

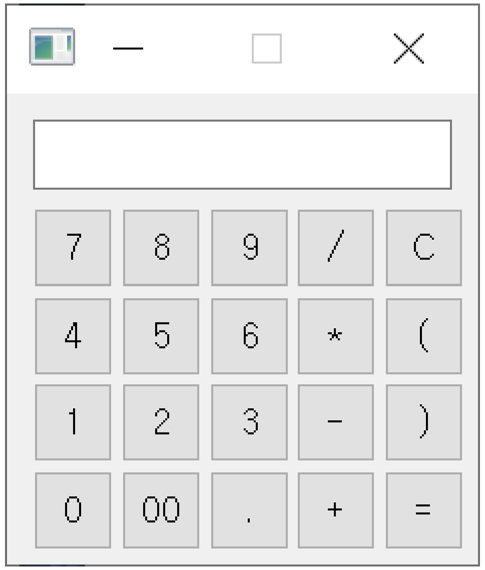
- 디스플레이의 고정 높이는 35 픽셀 이다.
- 디스플레이에 텍스트가 왼쪽 정렬로 표시된다.
- 직접 편집하지 않도록 디스플레이가 읽기 전용으로 설정되어 있다.

뷰 완성

```
def _createButtons(self):
       self.buttons = {}
       buttonsLayout = QGridLayout()
       buttons = {
                  '7': (0, 0), '8': (0, 1), '9': (0, 2), '/': (0, 3),
                  'C': (0, 4), '4': (1, 0), '5': (1, 1), '6': (1, 2),
                  '*': (1, 3), '(': (1, 4), '1': (2, 0), '2': (2, 1),
                  '3': (2, 2), '-': (2, 3), ')': (2, 4), '0': (3, 0),
                  '00': (3, 1),'.': (3, 2), '+': (3, 3), '=': (3, 4),
                 }
       for btnText, pos in buttons.items():
           self.buttons[btnText] = QPushButton(btnText)
           self.buttons[btnText].setFixedSize(40, 40)
           buttonsLayout.addWidget(self.buttons[btnText],
                                   pos[0], pos[1])
       self.generalLayout.addLayout(buttonsLayout)
```

뷰 완성

```
def setDisplayText(self, text):
        """Set display's text."""
        self.display.setText(text)
        self.display.setFocus()

    def displayText(self):
        """Get display's text."""
        return self.display.text()

    def clearDisplay(self):
        """Clear the display."""
        self.setDisplayText('')
```



1. .setDisplayText() 디스플레이 텍스트 설정 및 업데이트 한다.
2. .displayText() 현재 디스플레이의 텍스트를 얻기 위해 사용한다.
3. .clearDisplay() 디스플레이 텍스트를 지운다.

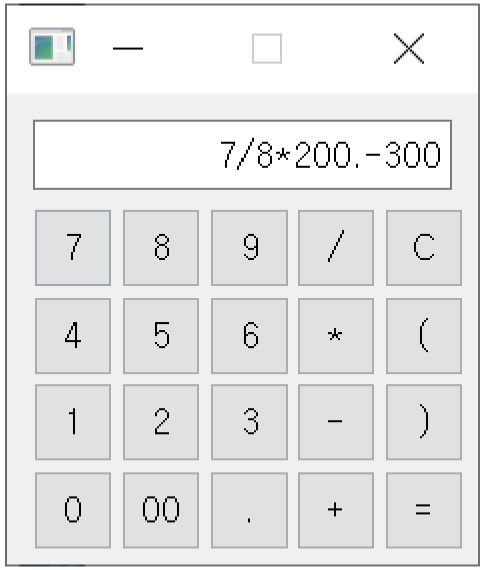
기본 컨트롤러 생성

```
class PyCalcCtrl:
    def __init__(self, view):
        self._view = view
        self._connectSignals()

    def _buildExpression(self, sub_exp):
        expression = self._view.displayText() + sub_exp
        self._view.setDisplayText(expression)

    def _connectSignals(self):
        for btnText, btn in self._view.buttons.items():
            if btnText not in {'=', 'C'}:
                btn.clicked.connect(partial(self._buildExpression,
                                            btnText))

        self._view.buttons['C'].clicked.connect(self._view.clearDisplay)
```

1. GUI의 공용 인터페이스에 액세스
2. 수학 표현식 작성 처리
3. clicked적절한 슬롯으로 버튼 신호 연결

기본 컨트롤러 생성

```
def main():
    pycalc = QApplication(sys.argv)
    view = PyCalcUi()
    view.show()
    PyCalcCtrl(view=view)
    sys.exit(pycalc.exec_())

if __name__ == '__main__':
    main()
```



등호 (=)는 아직 작동하지 않는다.

이 문제를 해결하려면 계산기 모델을 구현해야 한다.

모델 구현

```
ERROR_MSG = 'ERROR'
def evaluateExpression(expression):
    try:
        result = str(eval(expression, {}, {}))
    except Exception:
        result = ERROR_MSG

    return result
```

eval() 문자열을 표현식으로 평가하는 데 사용 한다.

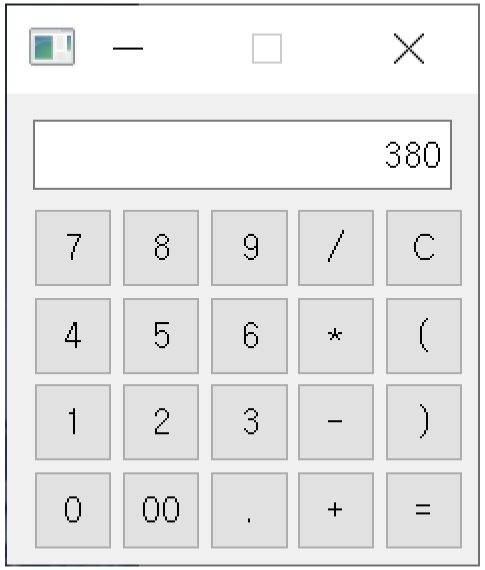
성공하면 결과를 반환한다.
그렇지 않으면 오류 메시지를 반환한다.

컨트롤러 완성

```
class PyCalcCtrl:
    def __init__(self, model, view):
        self._evaluate = model
        self._view = view
        self._connectSignals()

    def _calculateResult(self):
        result = self._evaluate(expression=self._view.displayText())
        self._view.setDisplayText(result)

    def _buildExpression(self, sub_exp):
        if self._view.displayText() == ERROR_MSG:
            self._view.clearDisplay()
        expression = self._view.displayText() + sub_exp
        self._view.setDisplayText(expression)

    def _connectSignals(self):
        for btnText, btn in self._view.buttons.items():
            if btnText not in {'=', 'C'}:
                btn.clicked.connect(partial(self._buildExpression, btnText))
        self._view.buttons['='].clicked.connect(self._calculateResult)
        self._view.display.returnPressed.connect(self._calculateResult)
        self._view.buttons['C'].clicked.connect(self._view.clearDisplay)
```

계산기 실행

```
def main():
    pycalc = QApplication(sys.argv)
    view = PyCalcUi()
    view.show()
    model = evaluateExpression
    PyCalcCtrl(model=model, view=view)
    sys.exit(pycalc.exec_())

if __name__ == '__main__':
    main()
```


380
7
8
9
/
C
4
5
6
*
(
1
2
3
-
)
0
00
.
+
=